THE EVOLUTION OF DEC'S PDP MINICOMPUTER ARCHITECTURE

HONORS THESIS

Pamela J. Kendrick

Advisor: Dr. Clinton Fuelling

August, 1982

Approved Clinton C. Fuelling 10/7/82

TABLE OF CONTENTS

APPENDIX

INTRODUCTION

Digital Equipment Corporation, abbreviated DEC, evolved in the 1950s as an electronics manufacturer. In 1957 DEC manufactured its first solid state modules; these were supplied to other computer manufacturers. Digital logic circuits were reduced, in 1958, to one circuit card, called a "systems module."

The PDP-1, DEC's first attempt at a computer, was developed in 1959. It was the first of its kind: a small, on-line computer. In 1963 the PDP-5 was introduced and stirred much public interest. However, its hand-wired components made mass production impossible. Flip Chip modules were manufactured for the first time in 1965 by DEC. These modules were fully implemented into internal logic, interfacing, and control components on their new PDP-8 system computer.

DEC has, since 1965, become the largest manufacturer of the mini-computer. DEC's success was largely due to the architectural structure and innovations made in both the PDP-8 and PDP-11 minicomputer families. Particular emphasis is placed on the evolution of architecture that has occurred for these two families. Also discussed will be five other less popular PDP families marketed. DEC systems, in general, exemplify high reliability, cost effectiveness, and an occurrence of timely technilogical innovations.

PDP-8

The PDP-8 was first developed in 1965. It was the first highly successful, mass-producible system manufactured by DEC. The PDP-8 was the first minicomputer, having both a low cost and small-scaled architecture. The PDP-8 could be used for general purpose applications; many were scientific-related. Low-level components consisted of TTL monolithic integrated circuit modules. The architectural structure present in all PDP-8 models enabled efficient processing and high system reliability. Major architectural innovations were made in 1970 with the development of the PDP-8/E and PDP-8/M models. Word length for all models was 12-bit.

PROCESSOR

The PDP-8 processor was the unit that controlled the interpretation and execution of instructions. Speaifically, the processor performed all arithmetic, logical, and system control operations. Around 1970, when the 8/E and 8/M models were introduced, a number of new registers, control circuits, and data gates and adders were integrated to enhance CPU performance. Systems configurations of 1965 and 1970 models are displayed in Appendix 1a.

Major Registers

A total of seven registers performed specific functions within the processor. The Accumulator was the only general purpose register available until the Multiply Quotient Register was added in the 8/E and 8/M models. The Program Counter was a control Register. The Memory Address Register and the Memory Buffer Register existed to aid in memory-CPU transfers. Data gates and adders were added in 1970 to provide for more efficient transfers. A Switch Register was included until 1970; this was operated on through the user's console.

The Accumulator (AC) was a 12-bit register upon which arithmetic and logical operations could be performed. AC contents could be cleared, complemented, and rotated left or right under program control. Contents could, as well, be added to MBR contents or combined by a logical OR with the switch register. The AC could, alternatively, be used as an I/O register; all information that transferred between memory and an external device passed through the AC.

The Multiply Quotient register(MQ) was a 12-bit, bidirectional, shift register that acted as an extension of the AC during Extended Arithmetic Element (EAE) operations. The MQ was integrated into the processor in the 8/E and 8/M models. During a multiplication operation, the MQ contained the multiplier at the beginning of multiplication and the least significant half of the product at the end. In the division process, the least significant half of the dividend and the quotient were held by the MQ. For shifts, the MQ contained the least significant half of the number. Moreover, the MQ was available as a temporary storage register, to provide more capability and flexibility for the user.

The Program Counter (PC) was a 12-bit register needed for controlling program sequences (the order of the instructions). The PC contained the address of the next executable instruction. Information entered the PC from the Memory Buffer Register and was then sent to the Memory Adress Register to locate the contents in memory. The PC contents were incremented after each word transfer to establish consecutive memory locations and to allow for the skipping of instructions based on test conditions.

The Memory Address Register (MAR) worked in conjunction with memory during instruction executions. This 12-bit register contained the memory address to be selected for reading or writing. All 4K words in memory could be addressed directly. The MAR was never cleared. Instead, after the existing address was located in memory, a new address replaced it. Data could enter the MAR from either the PC, the Switch Register, the Memory Buffer Register, or data break facilities.

The Memory Buffer Register (MBR), also a 12-bit register, temporarily held all information transferred between the processor and memory. The MBR could be loaded and incremented simultaneously before being read by memory. During data breaks, information was loaded from either an I/O device or memory. At other times data was loaded from either the AC or PC. The memory cycle speed (the memory cycle consisted of information being read from memory and then rewritten onto the same memory location) improved significantly between 1965 and 1970.

Data gates and adders were integrated into the Major Registers Module in 1970 for the purpose of moving data from one register to another more efficiently. Furthermore, a larger number of major registers were able to directly communicate with one another. Register information was gated as input to the adder. In turn, the shift gates output data to then be input to any of the major registers.

The Switch Register was included in all models up until 1970. This register allowed for information to be set into either the PC or the AC manually through the console keys.

Register Controls

Register Controls enhanced the performance of the major registers. Three registers were part of this control: the Link, the Instruction Register, and the Major State Register. An Instruction Decoder and major register control circuitry were also included.

The Link was a 1-bit register used as an extension to the AC. It was, in fact, a carry register for 2s complement arithmetic operations. Link contents could be cleared, complemented, or rotated with the AC contents. Furthermore, overflow for both single and double precision arithmetic calculations could be checked using the Link.

The Instruction Register (IR), a 3-bit register, held the operation code ("op" code) of the current instruction. The IR was loaded from the instruction during the fetch state. Contents of the IR were then decoded by an Instruction Decoder to perform the instruction function and to determine the proper states to enter.

A Major State Register controlled the processor and memory cycling. The four states (or cycles) were the fetch, defer, execute, and data break. Conditional inputs taken from each instruction determined the states entered. Each signal issued during state generations enabled corresponding register control circuitry. This register was added with the 8/E and 8/M models.

Finally, major register control circuitry was implemented into the processor at the time data gates and adders were introduced (1970). This circuitry controlled the enabling and disabling of the adder inputs and shift gate outputs. The circuitry also gated time pulses in order to cause the loading of appropriate registers.

Timing and Control

The timing and control module consisted of a number of elements. The Time Pulse Generator, register controls, and program controls comprised all models up to 1970. Interrupt control circuits and a Processor IOT Decoder were added in 1970.

The Time Pulse Generator accessed program instructions to determine which pulses were to be generated. The pulses output initiated timesynchronized gating operations. Up to 4 states were issued, and each new pulse was activated at the end of the previous pulse (a form of "pipelining"). The Time Pulse Generator also controlled memory timing.

Register controls were part of the gated inputs and outputs found in each of the major registers. In this way each register was tied to a common register bus. Data transfer was performed through both the bus and register input/output controls. Data was modified by enabling the adder/shifter and strobing the modified contents back to the original register.

Program controls were integrated for aiding input/output. When IOT (input/output transfer) instructions were executed, program controls produced pulses from an IOP (input/output pulse) generator. These pulses, in turn, initiated I/O operations. They also allowed peripherals to interrupt the processor for I/O service.

Interrupt control circuits were added to the Timing and Control module in 1970. These circuits comprised a major portion of the interrupt system. They responded when an interrupt request was issued.

A Processor IOT Decoder was also added in 1970. This component decoded the last nine bits of each IOT instruction. The proper I/O operations were thereby determined.

Major States

Major States consisted of the fetch, the defer, the execute, and the data break states. The former three were produced as a function of the current instruction. Data break states resulted from I/O device request signals. Each state occupied one timing cycle.

In the fetch state a 12-bit instruction was obtained from memory. The instruction then entered the MBR, the op code was passed to the IR, and the PC was incremented by one. If the instruction was a single-cycle type (augmented), the next cycle was another fetch, since operations were performed immediately. If the instruction was multi-cycle (a Memory Reference Instruction), the next state was either the defer or execute depending on the op code specified. The MRI instruction contained an address needed to find the proper operand. The major state register caused entry into the appropriate state at the end of each cycle.

The defer state was entered only when an indirect address was requested in a MRI instruction. Since the instruction contained the address of the address of the operand, memory had to be accessed a second time to obtain the second address.

The execute state was required for all MRI instructions. This made either the second or third time memory was accessed (depending on whether or not indirect addressing was specified). The operand itself was found in memory and the instruction was executed. The next state entered was the fetch.

The Word Count State, one of the data break states, was required only when a 3-cycle data break was requested. The word count in memory was transferred to the MBR and was then incremented. This word count was specified by the external device and contained the number of data break transfers to be enacted. The word count was repeatedly incremented by 1 for each data break transfer. Overflow occurred when the desired number of transfers had occurred.

The Current Address State was the second cycle required for a 3-cycle data break. This state established the data address to be transferred during the next cycle (the Break cycle). Transfers occurred at sequential addresses due to the incrementation done in the Word Count State.

The Break State was the final cycle of the data break states. It was required for both a 1-cycle and a 3-cycle data break (the third cycle in the 3-cycle break). The MBR temporarily held this data for both types of data breaks. The actual direct data transfer between memory and the external device was enacted during this state. The processor's major registers were not affected, although timing cycles were "stolen."

Arithmetic/Logical Unit Performance

Arithmetic/logical speeds are recorded in Appendix 2 .

Power Source

In general, no special source or environmental condition was required. A single power source of 115-volts and 60-cycles was needed. Both TTL (transistor-transister logic) monolithic integrated circuits and marginal error checking ensured reliable operation in temperatures between 32 and 130 degrees fahrenheit.

Options

Extended Arithmetic Element (EAE) enhanced arithmetic capabilities significantly. Operations were increased to much higher speeds because they were performed in parallel and on positive binary numbers. EAE contents included a Step Counter, an Instruction Register (for 4-bit op codes), timing and control logic, and mode flip-flops (to control the instruction set used). These components operated asynchronously with the CPU logic.

The MQ was an integral part of the EAE operations. It was used in conjunction with the AC (on 8/E and 8/M models) for multiplication, division, shifts, normalizations, and double precision arithmetic. Mode A was used in EAE operations when 3-bit op codes were specified. Mode B was required for 4-bit op codes. The 4-bit op codes (exclusively for EAE) enhanced double precision capabilities.

The Bootstrap Loader was a 32-word hardware ROM that was used as a Read-In-Mode paper tape loader. Programs were loaded to memory from the reader. User-defined functions could also be requested using the Bootstrap Loader. To operate the loader, the user depressed a console key and loaded the starting address. This operation actually started the computer.

Three different clock types were available.to perform timing synchronization. A line-frequency clock was a real-time, fixed interval clock. It interrupted 100-120 times per second depending on line frequencies. Crystal real-time clocks, also fixed-interval, interrupted either 50, 500 or 5000 times per second, depending on which number the user selected. Finally, a programmable real-time clock used IOT instructions to control clock operations. The user could, in this way, measure and count time intervals or events.

Power Fail Detect consisted of shut-down and restart subroutines. Computer operations were automatically restored when a primary power source failed. Low power was detected when the OMNIBUS caused an interrupt. All major register contents were retained.

MEMORY

Random-access magnetic core memory held 4K, 12-bit words. Memory worked in conjunction with the processor's MAR and MBR. These registers held memory address and transfer information for the processing of the next executable instructions. Locations 0 and 1 in memory were used during program interrupting. Locations 10 - 17 were reserved for auto-indexing. All other locations stored either instructions or data. Optional equipment provided error checking with a 13-bit word (parity bit included). Other options could extend memory capacities.

Features

Permanent information (longer than one instruction time) was stored and retrieved in memory until it was required by the processor. Memory itself did not perform operations on data. When reading or writing was required, data was transferred from memory to the processor in continuous cycles. Information transferring was done with circuits that performed the electrical conversions necessary to induce the transfers. Timing signals, issued by the processor, controlled memory. Memory cycle time improved a great deal between 1965 and 1970. For memory speeds, see Appendix 4 .

Standard features included: indirect addressing, internal facilities for performing instruction skips, and program interrupts. I/O address and data buffering were performed by processor registers.

Modularization

The PDP-8/E and -8/M models modularized their memory components. Three "quad modules" comprised memory. Extended memory was available in the form of 4K quad modules. Memory capacity was 32K.

The XY Driver and Current Source quad decoded address lines and drove XY wires. XY wires were driven for the following purposes: to decode addresses, to select switches, and to control XY current sources, stack discharge switches, and power on/off write protection. XY currents were controlled by the Sense/Inhibit module.

The Sense/Inhibit quad, the second of the three modules, contained the proper sense amplifiers, memory register, and inhibit drivers for 12-bit words. This module also controlled the XY current source and power supply for sense amplifiers.

Finally, the memory stack quad contained 4K words (the data and instructions) with a corresponding X and Y axis diode selection matrix.

Memory cores were actually mounted on a planar stack board for module assembly. This module had no direct connections with the OMNIBUS.

Address Setup

Memory was subdivided into addressable locations. Each word (data or instruction) was held in a 12-bit memory location. Each location corresponded to an 8-bit address. Up to 4,096 (4K) addresses could be used. 128 word locations comprised a block or "page" of memory. A 4K memory contained 32 pages. Either the current page (the page containing the instruction being executed) or Page 0 could be accessed by an instruction.

Each MRI instruction contained both the page and word address of the operand to be fetched. The page was indicated in bit 4. Bits 5-11 revealed which 1 of the 128 addresses on the page contained the operand. For a display of memory organization, see Appendix 5a.

Address Modes

Memory locations were accessed by 1 of 4 address modes. The four modes were the direct, the indirect, the indexed, and the program control modes.

Direct addressing was the most common mode. Bits 5-11 of each MRI instruction held an address to be found. The content of this address was the operand itself.

Indirect addressing was the second mode available. The address found in a MRI instruction referred to another address. The content of the second address was the operand. Memory was, thereby, accessed a second time. Indirect addressing also provided special advantages. Program control from one page to another and the return from a subroutine to a main program could be performed efficiently using indirect addressing. Furthermore, tables of data could be easily manipulated.

By using the indexed mode external events could be counted by a program. This count was stored in memory. It could be incremented by a certain sequence of commands. The counting was performed without disturbing the AC register contents.

The Program Control mode was a procedure of the PC register. The processor was directed by the PC to the next executable instruction. When the processor was ready for a fetch, the PC incremented itself. This directing and incrementing procedure could be changed somewhat. PC control could be transferred from one instruction sequence to another or could enter a subroutine.

<u>Options</u>

Memory Extension Control was available to expand memory capacities. Increments of 4K memory could be added, up to a total of 32K words. Addresses were extended from 8 to 15 bits when memory was enlarged (15 bits allowed for 32K addressable words).

In 1970 the Time Sharing Option was implemented into 8/E and 8/M models as an option. TSO ensured that a user's program was prevented from interfering with other users' programs.

Memory parity added circuitry to generate, store, and check 1-bit parities that were concatenated onto the 12-bit memory words. Thus, the 12-bit memory system was actually replaced with a 13-bit system. The parity formed for each word was retrieved and checked against its stored parity bit. If the two differed a parity error flag was set. This flag was connected to an interrupt to locate the interrupt source and enter the proper service routine. IOT instructions allowed the user to have some control over parity operations and the status flip-flop.

The Read-Only Memory (ROM), developed in the early 1970s, came in either 256 or 1,024 words. The information content desired had to be specified at purchase since it was wired at the factory before installation. The ROM was used in 3 ways: as a hardwired controller, for communications functions, and for process-controlled functions.

The Read/Write Memory was a 256-word module similar to the ROM but including write capabilities and protection. This module was configured with main memory and was required whenever a ROM was used. The Read/Write Memory was often used to simulate a ROM for program debugging and short-term storage before the information content was finalized. Other uses were for variable storage, interrupt handling, programming of constants, and protecting of user-monitored programs.

INPUT/OUTPUT

The flexible, high-capacity, I/O capabilities available on the PDP-8 enabled operations on a variety of peripherals. I/O processing required control circuitry within processor, memory, and external device interfaces. Information exchanges were made through a program (using the AC) or through a device (using the data break facilities). No modifications were required in circuitry when devices were added to the system. The OMNIBUS was added in 1970 to ensure better control and flexibility in information transfers.

Interfacing

Circuits in the processor, memory, and I/O devices enabled overall connection through a bus. Signals for data transfer were usually in the form of serial input pulses. Each I/O device had to detect its own select, code and provide the necessary I/O gating.

The main interface was the OMNIBUS. The OMNIBUS was integrated in the 8/E and 8/M system configurations. It was a back-plane, etched circuit board that connected all modules in the system. The OMNIBUS eliminated all back-plane wiring. Also every pin in a connector slot was defined to a particular device, thus, new external devices could be added by simply plugging them into the OMNIBUS. See Appendix 8a.

The OMNIBUS accomodated 96 signals that fed into 96 pins (the pins were located in connector slots). Signals controlled data transfers, addressed memory, and contained the actual data to be transferred. Two sides of the interfacing, the processor and the device interface control, shared common OMNIBUS lines. However, each line could only be used by one device at a time. The OMNIBUS lines carried either data, control, or data break information from I/O devices. An external bus was used in conjunction with the OMNIBUS.

The external bus was the second main interface. It consisted of a number of signal lines that enabled data and control information transfers. Until the development of the OMNIBUS, the external bus was the main means for information transfers.

The external bus became a supplement to the OMNIBUS in 1970. Two interface boards converted internal bus signals from the newer (8/E and 8/M) to earlier model signals. This conversion allowed the newer and earlier system components to be combined in one system.

I/O Units

The Automatic Send-Receive model-33 teletype was the standard I/O unit included with system purchase. An 8-bit ASCII code was used for character interpretation. Input forms available with the teletype were the keyboard and paper tape; output was in the form of either hardcopy print or paper tape. Spaces and marks on the paper tape corresponded to binary ones and zeros. Status flags were needed to aid reading and writing, since data transfers between the AC and device buffers were much faster that those between the buffers and the device itself.

Data was supplied as input to the computer through paper tape or keyboard. An 8-bit buffer contained in the keyboard or reader control was called the teletype input buffer, abbreviated TTI. The TTI held and assembled the last character struck. Data transfers were program controlled. Instructions could check the status flag for a ready condition. Another instruction then cleared the AC, cleared the flag, and transferred the data from the TTI to the AC.

Output was either punched onto paper tape or typed onto paper. A character was sent in parallel from the AC to an 8-bit buffer labelled the teletype output buffer (TTO). Data was then transmitted from the TTO to the output unit. When a status flag indicated a ready condition, the TTO was able to receive a new character from the AC.

The LA30 DECwriter, a dot-matrix matrix printer, was introduced in 1971. Its speed was 30 characters per second, three times the speed of the teletype. The DECwriter was actually a full-scale, hardcopy, I/O terminal that used a dot matrix to generate characters on ordinary paper (as opposed to electrostatic or thermal paper). Its quiet operation and high reliability were due to the substitution of solid state logic modules in place of mechanical parts.

Many other I/O devices were available as options. A high-speed paper reader and punch could enhance paper tape performance. Mass storage devices included magnetic tape, magnetic disk, and random-access disk. Less popular devices were line printers, XY plotters, Cathode-Ray Tubes, and card readers and punches. See Appendix 9a for I/O device transfer rates.

Programmed I/O Data Transfers

Most data transfers were performed with programming through the AC register. This type of transfer was much simpler and less expensive than data break transfers, but it required six times more CPU time. The programmed method was best when the processor's main purpose was to service I/O devices.

A program controlled both loading and storing. Gating circuits produced the proper pulses for the two operations through bussed connections. The program remained in a continuous waiting loop while waiting for a "ready" signal to perform the transfer. Simultaneous device operations were limited by device speeds and search time.

Three components were required for programmed data transfers. A device selector (bits 3-8 of an I/O transfer instruction) caused an I/O pulse response to the IOT command. Gating circuits, the second component, strobed both control and AC signals. These signals, in turn, determined the I/O bus requested and whether input or output was specified. The third component, busy-done flip-flops, pulsed the I/O bus, with the aid of gating circuits, to check the status flags for "transfer ready" conditions.

A standard process was followed for data input. When transfer status indicated a ready condition, a flag connected to an I/O skip (IOS) instruction was set by the device. The program then issued an IOT instruction in order to strobe the contents of the device buffer register into the AC by OMNIBUS lines. The transfer was actually performed by sending the data to memory and then to the AC. The program finally cleared the device or initiated further operations.

Output of data was accomplished in a reverse order. The AC contents were loaded from memory. An IOT instruction was issued to transfer a word to either the control or data register of the selected device. Another IOT instruction actually caused the operation to be performed on the device. When processing was completed (data transferred), the device status flag and external data register were cleared.

Programmed Interrupt Data Transfers

Since wait flags consumed much of the processor's time, the interrupt system was developed. Interrupts relieved the main program from the burden of repeatedly checking flags. The processor was able to ignore the peripherals until they requested interrupt service. The interrupt system had program implications, but consisted of hardware.

Control was transferred to a subroutine when an interrupt was issued by an IOT instruction. The subroutine first determined which device requested the interrupt and then serviced the device. This subroutine was found at location 0 in memory, and the service routine was in location 1. The processor automatically jumped to location 0 in response to any interrupt.

Data Break Data Transfers

The data break type of data transferring was performed without CPU intervention. I/O transfers were made directly with memory on a cycle-stealing basis. The major registers in the processor were, thus, left undisturbed. The external device itself controlled the data transfer. Data break devices required more control logic circuits and were more expensive. Data breaks were well-suited for large block transfers and high-speed devices. Up to 12 data break devices could be connected to the OMNIBUS.

Data break facilities in each device consisted of a number of components. Earlier versions used hard-wired flip-flops for information storage. Major parts were direction, break request, and cycle select flip-flops. The 8/E and 8/M models replaced the flip-flops with registers to increase efficiency. A Current Address Register, Word Count Register, I/O Buffer Register, and Break Priority Register comprised the 8/E and 8/M data break facilities in each data break device.

Either a 1-cycle or 3-cycle data break could be requested. A 1-cycle break only used 1 cycle for memory access. Since the word count and current address contents were stored in the device's registers, no extra cycles were needed to search memory for this information. The 3-cycle break consisted of 3 cycles of memory. The word count and current address were stored in memory locations, and this fetching required 2 cycles.

Data break requests were followed by a standard flow of events. In the first step, the initial set-up, the proper memory addresses of the word count and current address were loaded for 3-cycle breaks, and the word count and current address registers were initialized for 1-cycle breaks. In addition, this data break subroutine generated IOT instructions to enable the device control logic and to specify the direction of transfer. If a 3-cycle break was requested, one cycle was used to fetch the word count from memory and check for an overflow condition. If overflow occurred, the device control logic would be disabled following the current data transfer. In the second cycle (only for 3-cycle breaks), the current address was accessed in memory. Finally, for both 1 and 3-cycle breaks, data was transferred from the MBR to the device data register. This cycle was also called the Break State. When the word count had overflowed and the transfer had been completed, a flag was set that indicated the transfer was done. The device exited from the data break state.

INSTRUCTION SET

The PDP-8 instruction set was divided into 2 broad groups. Memory reference instructions accessed memory to obtain an operand on which the operation was performed. These were one-address instructions. Augmented instructions, zero-address type, consisted of all instructions not referencing memory. Augmented instructions issued control commands and I/O operations. The PDP-8 instruction set was extended between 1965 and 1970 to include 40 basic instructions. Other instructions were available with optional components.

Memory Reference Instructions

Memory-reference instructions were used when an operand had to be accessed in memory. Either two or three memory cycles were required, depending on whether indirect addressing was specified. The fetch cycle located the instruction (containing the operand address) in memory. The defer cycle, required if indirect addressing was used, consisted of accessing the second address in memory. In the execute cycle, the operand was fetched from memory and the operation specified in the op code was performed. The instruction format, displayed in Appendix 10a, included an op code, indirect address flag, memory page, and address. Basic cycle time was 1.5 microseconds for direct addressing and 3.0 microseconds for indirect addressing.

Augmented Instructions

Augmented instructions did not reference memory for operand accessing. Only one cycle, the fetch cycle, was required to fetch the instruction and initiate operations. Augmented instructions were also unique since each instruction could enact sequences of operations, a type of microprogramming. Each bit of the instruction corresponded to a certain operation. Thus, multiple operations could be specified in one instruction and were performed in a predefined sequence. Two subgroups of the augmented type were the I/O transfer (IOT) group and the Operate group. See Appendix 10a for formats.

IOT instructions initiated operations on peripherals and effected information transfers between the processor and I/O devices. Furthermore, interrupt instructions could request the interrupt facility. The interrupt system could also be turned on or off through programmed control.

The Operate instructions were further divided into Groups 1 and 2. Group 1 performed such microoperations as a clear, complement, rotate, and increment on the AC register. By using Group 2 instructions, a wide variety of operations could be specified. The AC contents could be checked, operations could be continued, instructions could be skipped, and other control commands could be issued.

MAJOR CHANGES ON THE PDP-8 FAMILY

Processor

- More register controls and timing generator elements were added to the CPU.
- The MQ register was implemented to increase programming capabilities, especially with arithmetic calculations.
- Decoders, multiplexors, and other control features were used in circuitry, enhancing versatility and capabilities within the processor.
- Processor components were modularized.
- The Time Sharing Option supported multiple users.
- Arithmetic calculation speeds increased significantly.

Memory

- Memory transfer speeds improved.
- Read/Write and Read Only Memories were introduced.
- Memory components were modularized.

Input/Output

- The OMNIBUS was developed, increasing data transfer control.

Instruction Set

- The instruction set was extended.
- Flag instructions for I/O devices were introduced.
- Many EAE instructions were added.

PDP-8 MODEL HISTORY

1965 The first PDP-8 was developed. It was actually nearly identical to the PDP-5 except a mass-producible version. Main purposes included dedicated laboratory and dedicated process-control applications. The small scale and low cost led to its labelling as the first minicomputer. The PDP-8 was remembered as: being the first mass-produced computer, the first popular minicomputer, and the first computer selling for less than $20,000.

1966 The PDP-8/S ("S" for serial) was the first OEM computer. The 8/S integrated serial rather than parallel logic. Thus, instruction executions were 5 times longer than on the 8 model, but the price was lower.

1967 The PDP-8/I ("I" for integrated circuit version) incorporated TTL logic into its processor for a more compact, economical, and rugged system that was easy to interface and maintain. The machine was fully parallel and allowed for easy and economical additions of I/O devices.

1968 The PDP-8/L ("L" for low) was an OEM version of the 8/I. It was the lowest cost full-scale computer offered. The 8/L was limited on speed, options available, memory capacity, and I/O configurability.

1970 The PDP-8/E ("E" for Expanded) was an expanded and much more sophisticated model. The 8/E was designed, as earlier models, for general purpose needs. It was fast, compact, inexpensive, and easy to interface in comparison to previous models. Modular expansion for specific applications was available. Among the technilogical innovations added was an OMNIBUS for effective I/O device communication.

1970 The PDP-8/M was an OEM version of the 8/E. The 8/M was identical in performance capability to the 8/E, except that I/O configuration was limited.

1974 The PDP-8/A was developed for OEM use and for systems builders. A more conventional configuration set was used for memory, the processor, and the power supply. Performance fell between the larger, 16-bit systems and the smaller microcomputers. The limits of memory addressability (because of a 12-bit word length) have not been dealt with. The 8/A is the most popular model at the present time.

OTHER PDP FAMILIES

PDP-1

The PDP-1 was DEC's first computer, introduced in 1959. This closely followed DEC's development of the electronic circuit ("solid state") modules. These modules supplied other computer manufacturers.

The PDP-1 was the first of small, on-line computer systems. In fact, DEC introduced the small-scale computer concept. The PDP-1 provided the standard data processing functions available at that time. Furthermore, it could be connected to various instruments and equipment to provide on-line, real-time monitoring for control and analysis purposes. The PDP-1 sold for an economical $120,000, while other computer prices averaged $1 million.

PDP-5

The PDP-5, marketed in 1963, represented DEC's commitment as a computer as well as electronics manufacturer. The PDP-5 was manufactured with hand-wired production techniques, making mass production impractical. (The PDP-8 was a mass-producible version of the PDP-5). Words on the PDP-5 were composed of 12 bits. The PDP-5 was geared toward dedicated laboratory and dedicated process control applications. The average PDP-5 price was $28,500.

PDP-9

The PDP-9 was introduced around 1966. It surpassed the PDP-8 in handling on-line and real-time scientific applications requiring more computational power. Word length was 18-bit, which allowed for a significantly larger memory capacity (32K words). Add times and I/O transfer rates were significantly better on the PDP-9 than on the PDP-8. The PDP-9 was implemented for a variety of highly technical applications: biomedics, process control, chemical instrumentation, display processing, hybrid systems, data communication and physics.

PDP-10

The PDP-10, also developed around 1966, was an expandable system offering optimum power and versatility. Word length was an expanded 36 bits, thus, memory capacity was 262,144 words. A very powerful CPU contained 15 registers and 16 accumulators. Calculation speeds and I/O transfer rates were not quite as high as on the PDP-9. The PDP-10, an on-line and real-time system, was used for applications very similar to those of the PDP-9.

PDP-12

The PDP-12, created in 1967, was a laboratory system implementing the best features of DEC's LINC-8. (The LINC-8 was designed to control experiments, collect data, and analyze data in the lab. Both the PDP-8 and MIT LINC instruction sets were available for researchers). Applications included: biomedics, physics, chemistry, meteorology, oceanography, psychology, radiation, acoustics, and seismology.

PDP-11

The PDP-11 family of minicomputers, introduced in 1970, surpassed the PDP-8 by addressing its limitations. PDP-11 architecture implemented a more powerful processor, allowed for a much larger storage capacity, and performed more efficiently overall. The PDP-11 was originally developed as a result of the public's demand for a modular, real-time system that provided data transfer, control and analysis. The entire system was comprised of modular units. Other valuable architectural enhancements were added throughout the 1970s as new models within the PDP-11 family were developed. See Appendix 1b for a block diagram of the PDP-11 system.

PROCESSOR

The PDP-11 processor served the same purposes as the PDP-8 processor. It performed all arithmetic, logical, and system control operations in parallel. Word and register length was 16-bit, which extended the memory address length and capacity in comparison with the PDP-8. In addition, the CPU regulated all UNIBUS actions (the UNIBUS controlled all device-to-device communication). This regulation included handling data transfer and control commands. Four versions of the PDP-11 processor have been developed. Size decreased drastically from 19 boards in 1970 to 1 board in 1974. The overall goal within the PDP-11 family was to retain software compatibility across all models.

Major Registers

Unlike the PDP-8's one accumulator, eight registers were available to the user for general purpose processing. A program status word maintained the current status of program instructions. Many registers were internally implemented and unaccessible to the user. All registers were 16 bits in length.

General registers were all user-accessible. Most models contained 8 but a few systems contained more, as many as 16. Registers were used as: accumulators, index registers, autoincrement registers, autodecrement registers, stack pointers, and program counters. The seventh of the 8 registers was predefined to be a stack pointer for the memory hardware stack. The eighth register acted as the program counter by containing the address of the next executable instruction.

The Program Status Word (PSW) was a register that held the current status information of the processor and an executing program. The current processor priority and current and previous operational modes were contained in the PSW. The condition codes of the last instruction executed were also located in this register.

Many registers were labelled "non-accessible" for programming purposes. These were the processor's control and status registers. Registers aiding in CPU-memory transfers were the Memory Address Register, the Memory Buffer Register, and memory system error requests. Sets of registers were used for error detection and correction. Examples were the Error Registers, High Error Address Registers, Low Error Address Registers and Hit/Miss Error Regsiters. Finally, maintenance and control registers helped ensure smooth processor performance.

Timing

The Universal Asynchronous Receiver/Transmitter (UART) was the asynchronous system used for timing. The transmitter accepted characters in parallel, then output them in a serial, asynchronous fashion. The receiver working in reverse, input characters serially and asynchronously and released characters in parallel.

Major States

Instructions were processed more uniformly in PDP-11 systems than in PDP-8 systems. Each required at least 3 memory cycles, and could use more depending on the complexity of the address modes and on the type of operation. Instructions were processed asynchronously on a cycle-stealing basis.

The instruction cycle ("I-phase") was entered in order for the instruction to be accessed in memory. The instruction was also decoded for further processing. The operand cycle ("O-phase") located any operands needed for single and double operand instructions by searching in memory. The execute cycle ("E-phase") performed the actual operation specified in the instruction.

Unlike any PDP-8 instructions, a PDP-11 instruction could cause multiple O-phase and E-phase cycling. Multiple cycling was necessary, due to the existence of both double operand instrutions and the range of addressing modes available. Although instructions required more cycling, this was much less time-consuming overall and more efficient than executing more instructions. Separate cycles did not exist for data break transfers.

Arithmetic/Logical Unit Performance

See Appendix 2 for PDP-11 arithmetic and logical speeds.

Power Source

As in the PDP-8, 115-volts and 60-cycles of power were required for PDP-11 systems. If this supply was not maintained, a trap was initiated. The trap service routine was given 2 milliseconds to save all register contents and to condition the peripherals.

Microprogramming

Microprogramming was implemented in the processor control module in PDP-11 models. This was a very efficient and systematic method for generating processor-initiated sequences of control micro-operations.

Microprogramming, with its structured, hardware design, replaced hard-wired control units that used the conventional logic techniques.

Among the innovations developed in the 11/60 model was the availability of user-accessible microprogramming. The user could, in this way, customize the processor somewhat for performing specific needs.

Microprogramming solved any problems of incompatibility between older and newer systems. Three optional forms of microprogramming could be used. The User Control Store contained 1K of RAM and allowed both storing and writing . The Extended Control Store option offered 1,536 words of ROM for a microprogram. A Diagnostic Control Store consisted of a ROM allowing isolation and analysis of processor faults.

Data Formats

Word length in PDP-11 systems was 16-bit or 2-byte (8 bits comprised a byte). Bytes could be processed as well as words. A 2-bit parity extended word length to 18 bits. Instructions were labelled "variable length" since they could consist of 1,2, or 3 words (2 or 3 words are required for both the index mode and program counter modes). Floating-point operands, 32 bits in length, were divided into an 8-bit exponent and a 24-bit fraction. Double precision operands were 64 bits; 56 of these were the fraction bits.

Unique Processor Capabilities

Reentrant code, also labelled Position Independent Code (PIC), was a very valuable feature found on all PDP-11 systems. This code enabled 1 copy of a subroutine or program to be accessed by more than 1 process or task. Memory requirements were, thereby, reduced for multi-task applications. A relocating loader (or "linking loader") relocated programs into different memory locations each time they were executed.

As long as relative addresses remained identical, the same offsets could be used for relocations.

The "stack" was an area of memory set aside for temporary storage and linkage. The seventh general purpose register was predefined to be a hardware stack pointer. It was automatically implemented during subroutine linkage and interrupt service.

Coroutines were alternatives for subroutine use. Coroutines referred to multiple program segments or routines that were interactive, but no routine was subordinate to another. Control was passed back and forth between coroutines, and each routine was "suspended" between executions.

Interrupts were similar to subroutine calls, except that they were controlled by hardware. Since the processor was required only upon device request, it could spend less time in wait loops and more time in actual processing.

Recursion was a feature using the stack facility. A program was able to call itself using recursion, just as it would call a subroutine. The stack stored intermediate variables and linkage information associated with each recursive call.

Traps were actually interrupts generated by software. Errors and programming conditions could automatically respond with certain trap routines. The priorities of executable traps were (from highest to lowest) bus errors, instruction traps, trace traps, stack overflow traps, and power failure traps. For other trap information, see Appendix 3 .

Technilogical Changes

In 1970, the first systems (the 11/15 and 11/20) contained processors consisting of 19 boards and hundreds of integrated circuits. A second processor was developed in 1971 and had tremendous improvements implemented in its architecture. This CPU, found in models 11/45, 11/05, and 11/40, used the modern Schottky TTL logic and consisted of only 2 boards. A new "cache" bipolar memory was integrated as well as a new "MOS" (metal-oxide semiconductor) memory. Both memories, though more expensive than core, greatly enhanced processor performance and speed..

The 11/04, introduced in 1974, integrated MSI (medium-scale integration) components to reduce the processor size further to 1 hex board. This new processor had the flexibility to accept either core or MOS memories. A new power supply was developed internally. Consequently, performance levels increased by 15 percent.

The 11/60 implemented a fourth processor. Microprogramming was made accessible to the user. A hardware multiply and divide were standard features. Furthermore, single and double precision floating-point arithmetic were standardized.

<u>Options</u>

A Floating-Point Processor (FPP) made 64-bit operations and 17-digit precision available (8-digit precision was standard for 32-bit operations). Arithmetic operations were calculated much faster using the FPP. The FPP was able to reference any memory location or processor register. It also used address modes and memory management similar to those implemented by the processor. Finally, 6 64-bit floating point registers were available.

An Extended Instruction Set allowed hardware, fixed-point multiply, divide, and shift operations to be performed in general purpose registers. Double precision capabilities were also included.

A Commercial Instruction Set provided up to 10 extended instruction groups. Two types of groups, register and on-line, determined how operands were to be delivered to an instruction. This option was implemented many times for enabling faster COBOL executions.

Error Correction Code (ECC) was a feature included in MOS memories whereby certain bits were stored in a word in a unique combination. The words were later checked for errors before being sent from memory to the processor. Errors that were detected were also corrected by ECC. ECC provided the best security for multiple, single-bit failures, the most common error type in MOS memories.

The bootstrap loader, very similar to that in the PDP-8, was required to initially load a blank tape, a sector of disk, or other device. A bootstrap program loaded the status register of the device specified.

Three clock types were available for timing synchronization. These included real-time, line real-time, and line-frequency clocks. These clocks were very similar to those discussed in the PDP-8 processor options.

MEMORY

Perhaps the most significant developments made in the PDP-11 family were with memory. Main memory was designed to handle both bytes and words. Basic memory space in all PDP-11s was 32K words. The upper 4,096 words were reserved for I/O control and buffer registers (the "I/O block"). The lower 255 words were reserved for interrupt vectors, trap vectors, and floating vectors. The 11/70 used interleaving to improve processor-memory transfer rates. Interleaving consisted of organizing consecutive memory locations in different memory modules. Error detection was aided by both the parity and ECC features.

The invaluable memory management system, added in 1971, enabled the start of virtual addressing so that memories could be expanded without increasing address length. Virtual addresses were "mapped" onto physical addresses. The mapped memory feature limited the address space used by a program. This security measure ensured that several programs would not intrude with each other.

Main Memory Types

Three main memory types were available. They were core, MOS, and cache memories. These types, with their very different speeds, were an alternative method of establishing a memory hierarchy. In all cases, the higher the memory speed, the higher the cost of the memory type. See Appendix 4 for a summary of memory speeds.

Core memory was the only type of main memory implemented until 1971. Its non-volatility and low cost were particularly attractive features. The core speed was the slowest of the 3 types.

In 1971, MOS memory was introduced as a substitute for core memory. Though more expensive, MOS offered better performance levels and speeds. MOS was, in turn, slower than cache (the third type), but was more economical to manufacture and operate. Because MOS was volatile, backup batteries were needed to prevent data loss.

Cache memory was a very high speed memory that was usually used in conjunction with either MOS or core memory. Cache, or bipolar, memory increased processing speeds by making data available to the CPU at a rapid rate. Since main memory normally slowed the processor down, cache was very effective. Main memory contents were copied as they were required for processing. A cache "hit" sent the information to the processor.

In 1975, cache was placed between the processor and main memory for critical, intermediate storage. A FASTBUS was also developed especially for processor-cache memory transfers, allowing the UNIBUS to be bypassed.

Memory Management

The important development of the memory management system came about in 1971. This system allowed memory capacities to be expanded and introduced virtual addressing. It was actually part of the operating system to supervise data transfers between memory devices (main memory, tape, disk, drum, etc.). The main objectives of the memory management system were to maximize computer component utilization and to allow for expandable memories without enlarging memory addresses.

The direct address was considered to be a virtual address rather than a physical one. Through the mapping process, the processor could access more than 32K words. Three forms of memory mapping were available. These were 16-bit, 18-bit, and 22-bit systems. See Appendix 6 for a diagram. The relocation of programs by memory was transparent to both the user and the program through mapping onto auxiliary storage devices (disk or tape).

The physical address was extended in the following way. A virtual address was combined with a corresponding Page Address Register (containing relocation information) to yield a physical address. The allocation of physical memory was provided for all programs. System and user functions were physically separated. See Appendix 7 for a page mapping illustration.

Three modes of operation were selectable. Each mode had its own set of active page registers and could access up to 8 pages of data and instructions (each page could range from 64 to 8,192 bytes). The Kernal mode enabled a program to map a user's program anywhere in memory (this program was controlled internally in the system). Moreover, users' programs, related registers, and the PSW contents were protected from the surrounding environment.

In both the User and Supervisor Modes, the user assumed control (versus the system program). In the User Mode, programs could be prevented from modifying machine states relating to memory mapping and protection. In the Supervisor Mode, a control program state was provided to help make system management more efficient and secure.

Addressing

Memory reference addressing was accomplished using the 8 general purpose registers. These registers (instead of instructions, like in PDP-8 systems) contained addresses and operands as intermediate storage so that processing could be done faster. Each memory reference instruction specified the op code, the register(s) to be used, and the addressing modes (how the register was to be used). The variety of addressing modes available extended instruction set capabilities to provide for very efficient and flexible data handling. The complexity of the addressing modes reduced the number of instructions required to perform operations. Instruction formats were variable length in order to handle index mode and program control modes.

Both byte and word addressing could be specified. Words were assigned even-numbered addresses. Five types of transfers were addressable: register to register, register to memory, memory to memory, memory to stack, and register to stack.

Address Setup

As previously mentioned, basic memory space consisted of 32K words. The upper 4,096 words were predefined for I/O device registers. The lower 255 words were reserved for various vectors. For a diagram, see Appendix 5b.

Address Modes

Three address mode types could be specified on all PDP-11s. These included the direct, indirect, and program counter mode types. Sets of three bits in each instruction contained the address mode and the register involved in searching for each operand. Addressing modes handled 3 types of addressing: two-address (for double operands), one-address (for single operands and branches), and zero-address (for control commands).

Each type of addressing had four individual modes available. Direct addressing modes included: register, autoincrement, autodecrement, and index. Indirect addressing modes were comprised of: register deferred, autoincrement deferred, autodecrement deferred, and index deferred modes. The index and index deferred modes used a second word in the instruction that was combined with the PC to form effective addresses. Program counter addressing consisted of: immediate, absolute, relative, and relative deferred modes. Each of these modes required either 2 or 3 word instructions to contain an operand, operand address, relative address, and relative deferred address, respectively.

INPUT/OUTPUT

A drastic change in input/output features took place with the introduction of the PDP-11 family. The UNIBUS was the means of communication between all system components for program-controlled data transfers, data break transfers, program interrupts, and priority bus control. The UNIBUS also permitted a unified addressing structure so that peripheral device registers could be addressed directly as memory locations. By using all memory reference instructions on device registers, I/O programming had increased flexibility. Peripherals became programmable instead of being accessible only be special IOT instructions. Finally, the video and hardcopy terminals replaced the teletype as the main I/O device.

UNIBUS

The UNIBUS, the most distinguishing and unique feature on the PDP-11s, was actually patented by DEC in 1970. It was a single, high-speed, asynchronous, bidirectional communications path that enabled components to communicate independently of the processor and without intermediate memory buffering. The UNIBUS was the first data bus in minicomputer history to provide these capabilities.

The UNIBUS actually consisted of a group of wires that performed simultaneous data transfers. A set of 56 signals comprised the bus; 51 of these were bidirectional signals and 5 were unidirectional. This single set of signals ("single bus concept") ensured better overall control and easier source-destination interrogations. Each device had its own signals. Two were required, one for actual data and one for control information.

I/O devices were addressed, as mentioned before, by the same method as memory. Specific memory addresses were reserved for only I/O use. Instructions processed on device registers, thus, no intermediate device data buffering in memory was required. Each device contained its own control, status, and data registers. It also had a controller for reading or writing to memory (just as the processor did). The UNIBUS had the capacity to accept a large number of I/O devices. Devices were merely plugged into the UNIBUS to become active.within the system.

A "master/slave" relationship existed in component communication. One device always had control over the bus; it was labelled the "master." The master commanded the bus to communicate with the receiving device, referred to as the "slave." The master/slave communication was dynamic. In other words, control was easily passed from device to device. All components had the capabilities of acting as either the master or the slave.

Related to the master/slave feature was the "bus interlocked" communication process. Each signal released from the master had to be responded to by the slave, or information transfer would not occur. Physical bus length and master/slave response time had, however, no effect on bus communication. One word was transferred per bus cycle. An instruction cycle consisted of one or more bus cycles.

Bus Requests (BRs) were issued when a device needed to interrupt the processor for performing a service routine for the device. The BR was not serviced until the processor completed its current instruction processing. Interrupt handling was automatic, and device polling was not required.

Bus Requests followed a certain sequence of steps. A device first issued a BR. A "Bus Arbitrator" permitted the device to take control only if its priority was highest. If so, the Arbitrator issued a Bus Grant (BG). The device received the BG, inhibitted further grants, and cleared the BR. The Arbitrator, in turn, cleared the BG. The device then asserted the Bus Busy Signal (BBSY) and cleared the inhibit signal. Finally, the device actually issued the interrupt and its vector address.

The processor responded to the interrupt in the following way. It was directed to the proper service routine by the given vector address. The current PSW and PC were stored onto the stack at the time of the interrupt request. New PSW and PC contents were taken at the vector interrupt address. Nesting of interrrupts was possible at any time after the PSW and PC were loaded. Upon completion, the old PSW and PC contents were restored. During an interrupt process, the processor and device always took on the master/slave role, the processor always being the master.

Non-processor requests (NPRs) were issued for direct memory access without processor intervention. The processor was able to relinquish bus control during NPR executions. NPRs had the highest priority on the UNIBUS.

A similar sequence of steps was required for the NPR executions that was followed by Bus Requests. The device issued a NPR. The Bus Arbitrator responded with a non-processor grant (NPG). The device accepted the grant, inhibitted further grants, and cleared the NPR. The Arbitrator received the inhibit and cleared the NPG. The device issued, at this time, a Bus Busy (BBSY) and cleared the inhibit. Data transfer then took place.

A priority system was implemented into the UNIBUS processing. Priority circuits decided which device became master when simultaneous UNIBUS requests occurred. Each device was assigned a priority level at the time of system installation. Priority assignments were based on three factors: the device's operating speed, service requirements, and data recovery abilities.

The priority control structure, displayed in Appendix 8b, consisted of 2-dimensional (vertical and horizontal) levels. The five main levels, or lines, were vertical. If two devices with equal priority levels (on the same vertical line) requested the bus simultaneously, the device electrically closest to the processor would gain control of the bus. After completion of a request, the device with the second highest priority was given control. Highest priority levels were assigned to high-speed devices, to prevent data loss. The processor was given lowest priority, since interruptions caused less serious consequences.

The NPR line, connecting all direct memory access (data break) devices, was the highest priority line. The non-processor grant (issued by the Bus Arbitrator) was passed from device to device on the NPR line until the one that requested control was found. This device then took command on the NPR line.

BR lines were on 4 vertical levels. Higher speed devices were placed on higher vertical lines. Each BR line had its own BG chain. Requests made on the same BR line were granted to the device electrically closest to the processor. 4 other interrupt lines, lower than both the NPR and BR lines, were not assigned devices. These levels were used by software only.

MASSBUS

The MASSBUS was first introduced in 1971 (the 11/45 model) as a UNIBUS extension for interfacing the new bipolar and MOS memories. The MASSBUS especially enhanced faster I/O transfer performance.

Interfacing

Interfaces were required in all I/O devices for efficient UNIBUS processing. Interface controllers supplied the data communication control information for a system. Four communication channels were available: the private phone, the dial-up phone, a telegraph line, and modems. A variety of hardware interfaces were offered for the different I/O transfers types.

Programmed I/O

All manipulations for I/O operations were, as mentioned previously, performed with memory reference instructions. I/O programming was, thus, much more flexible than that done on the PDP-8. Because of predefined memory locations and device registers, all information could be directly compared. No intermediate registers were needed for comparisons.

All external devices required at least two registers. The first register contained both control and status information. A second held the actual data. More complex devices could have multiple registers.

Control and Status Registers contained all information necessary to communicate with that device. At least one was assigned per peripheral. Control/status bits held such information as: the device unit number, the device's function, current conditions, error enable status, and done enable status. Data Buffer Registers transferred information that was travelling into or out of the computer. The number and type of data registers was a function of the device.

A terminal/keyboard reader was an example that described the interaction of device registers during programmed data input. The data was held (a character at a time) in a buffer register called the Terminal Keyboard Buffer that was contained within the device interface. A second register, the Terminal Keyboard Status register, held read busy, read done, and reader interrupt status information. The programmer constructed a wait loop in the program to check the TKS for when a character was available in the buffer. When one was ready, the character was moved from the TKB to the user-specified register.

I/O Devices

The types of I/O devices commonly used for user-accessible inputting and outputting changed greatly between 1970 and 1980. In 1970 the teletype, along with high-speed perforated tape readers and punches, was standard with all PDP-11 models. In 1980 video terminals and the DECwriter hardcopy terminals were the most popular and most efficient. Sophisticated line printer subsystems, high volume disk packs, magnetic tape subsystems, and other mass storage units were offered throughout the 1970s. See Appendix 9b for an I/O device summary.

INSTRUCTION SET

In general, all instructions were memory reference type and contained 16 bits per word. Either zero, one, or two-address instructions could be specified. Each operand was located through both an address mode and a register. The address mode and register combination increased programming efficiency and flexibility. Instruction specifications included an op code, a general register, and an address mode that specified how to access the operand. All systems had over 400 instructions in 1980; this was a tremendous increase over the 66 instructions available in 1970. All instructions were divided into 7 classes. See Appendix 10b for class formats.

Instruction Classes

Single operand instruction length was either one or two words. Two words were needed to contain the following index and program control mode information: an index (that formed the effective address for the operand), an operand, an operand address, a relative address, or a deferred relative address. 6 bits held the register and address mode of the destination field, and 10 bits held the op code. Examples of single operand instructions included the clear, complement, increment, shift, rotate, and multiple precision operations.

Double operand instructions first became available for PDP-11 systems (they were non-existent in the PDP-8 family). These instructions could be up to 3 words in length since they contained both source and destination operands. Double operand instructions increased the programmer's flexibility. The format consisted of 2 6-bit areas containing the registers and address modes of both a source and destination operand. 4 bits held the op code. Add, compare, move, and logical functions were examples of double operand operations.

A third class of instructions was the branch type. The instruction included an 8-bit op code (a condition to be tested) and an 8-bit word offset. The offset determined the branch target address relative to the PC. It was calculated using the following formula:

WORD OFFSET = (TARGET ADDRESS - BRANCH INSTR ADDRESS - 2)/2.

The offset was then truncated to 8 bits. Unconditional, signed conditional, and unsigned conditional branches were performable.

The subroutine class of instructions provided linkage operations for subroutine entries and exits. The seventh register was predefined as both the link and destination register during subroutine executions. 6 bits contained destination field information (register and address mode), 8 bits held the op code, and 3 bits held the linkage pointer register. Examples were the jump, jump to subroutine, and return from subroutine instructions.

Operate instructions were zero address type, thus, the format consisted of only a 16-bit op code. These instructions performed various control operatione, for instance processor halts, resets, and no-operations. Furthermore, a move could be made to a previous data space.

The condition code class of instructions enabled the user to access the contents of the PSW. Bits 5-15 held the op code, bit 4 indicated a set or clear operation, and bits 0-3 were the mask field (each bit corresponded to the 4 condition code bits in the PSW). Bits 0-3 reflected the contents of the PSW. The processor used this information to test conditions in programs. PSW bits could be checked for status or could be set or cleared by certain instructions.

Traps and interrupts were also executable by issuing certain instructions. These were zero-address type, and contained a 16-bit op code. The main purpose for these instructions was for exiting from a program. Exiting could be a result of a subroutine call or a result of an error or fault. A trap exit consisted of internal hardware forcing entry into a trap routine. A software exit was specified in a program by a jump instruction. In the final method, the interrupt exit, external hardware forced entry into a service routine.

MAJOR CHANGES ON THE PDP-11 FAMILY

Processor

- Overall performance of the processor greatly improved through more powerful and efficient processing capabilities and component interaction.
- The CPU size was drastically reduced through four versions from 19 to 1 integrated circuit board.
- Processing speeds improved for arithmetic calculations and data transferring.
- The implementation of microprogramming in control logic enhanced control capabilities. The later availability of user-accessible microprogramming provided increased flexibility.
- The Floating Point Processor contained sophisticated modifications in later models.
- CPU optional enhancements included: an extended instruction set, a commercial instruction set, and error correction code.

Memory

- Memory management provided an extremely efficient means of expanding memory capacities and capabilities.
- Cache and MOS memories enabled much faster I/O-memory and CPU-memory transfers.
- A FASTBUS was implemented to speed up Cache-CPU information transferring.

Input/Output

- User I/O devices have changed from the teletype to more powerful video and hardcopy terminals.
- Mass storage devices were modified to perform at higher speeds and more effectively through memory management.
- A MASSBUS was included in certain models for faster I/O transfers.

PDP-11 MODEL HISTORY

1970 The PDP-11/20 was introduced. The 11/20 was a modular, real-time system that performed data transfer, control, and analysis functions. The larger 16-bit word length enabled larger memory capacities than with the PDP-8 models.

1971 The PDP-11/15, an OEM version of the 11/20, was developed.

1971 The PDP-11/05 replaced the 11/20 and 11/15. This model included DEC's first major updates to the 11 family. The unit was repackaged for better cost-effectiveness and was an OEM model.

1971 The PDP-11/45 was similar to the 11/20, but contained extended capabilities. The 11/45 was a core-only machine. Memory management was integrated for the first time, a major enhancement to the systems, and raised main memory from 56Kb to 248Kb. Schottky TTL logic was incorporated into a new processor. A MASSBUS was included for faster I/O transferring. Two new types of main memory, MOS and Cache, were developed during this time period, but were unavailable to the 11/45.

1971 The PDP-11/50 was identical to the 11/45, except that it contained MOS memory instead of Core.

1972 The PDP-11/40 was very similar to the 11/45 and 11/50, and was, in fact, labelled a "fill-in-the-gap" model. Memory management was identical, 248Kb core memory was available, and integer operations were performed at twice the speed of the 11/20.

1972 The PDP-11/35 was an OEM version of the 11/40, though memory capabilities were extended.

1974 The PDP-11/04 contained many technilogical innovations that improved system price and performance. A new, low-end processor was incorporated. MSI components reduced the CPU size from 2 to 1 board. Architecture could accept either Core or MOS memory. Other improvements included: a new power supply, less expensive memory modules, and 15% greater performance levels overall.

1975 The PDP-11/70 was introduced for very large system use. LSI was incorporated in the processor. For the first time a 32-bit Cache memory was placed between main memory and the processor. A new FASTBUS connected the two.units. Because of memory management, 4 million bytes were addressable. The 11/70 had 75% of throughput capabilities of the IBM 370.

1975 The PDP-11/55 was essentially a PDP-11/45 specialized for scientific applications. Faster operation was due to 3 features: use of Cache memory, a unique 11/70 microcode, and a new and faster Floating Point Processor.

1976 The PDP-11/03 was an OEM model for single-terminal use or as a small distributed processing system.

1976 The PDP-11/34 was an outgrowth of the 11/04. Cache memory was available and memory capabilities were extended to 248Kb.

1977 The PDP-11/60, labelled a "FORTRAN machine," had capabilities between the 11/34 and 11/70 models. Memory features included an expandable memory, memory relocation, and memory protection. Microprogramming with user accessibility was another significant enhancement. I/O-memory and CPU-memory transfers could be performed simultaneously. A special diagnostic module was included in the processor, and a hardware multiply, divide, and single and double-precision floating-point arithmetic became standard features. A Floating Point Processor Instruction Set was made an available option.

1979 The PDP-11/44 was a midrange model. Some 11/70 features were available. Performance was twice that of the 11/34. One megabyte of ECC MOS memory and 8Kb of Cache memory was included. Options offered were: an Extended Instruction set, a Commercial Instruction Set, and a microprocessor-controlled programmer's console.

GENERAL COMPUTER ARCHITECTURAL TRENDS, 1965-1980

Processor

- Integrated Circuits replaced discrete transistor circuits to substantially reduce system size and cost.
- Small, Medium, and Large Scale Integration were integrated into processors.
- Microprogramming became widespread for better control unit operations.
- Techniques for concurrent or parallel processing were developed to increase execution speeds. Examples included: pipelining, multiprogramming, and multiprocessing.
- Time Sharing and Position Independent Code were implemented to improve resource utilization.
- The use of a Program Status Word became common.
- Microprocessors were used as distributed, dedicated processors, an alternative to the time sharing option.

Memory

- Semiconductor memories (MOS and Cache) replaced core memories.
- Memory management became a common addition. Memory space was, in this way, allocated dynamically in both main and secondary memories.
- Position Independent Code better utilized memory space.

Input/Output

- A single bus was often integrated for component communication and to access I/O devices like memory.
- Direct memory access between memory and external devices relieved the processor of many duties and much time.
- Microprocessors were used as I/O device interfaces for added control.

Instruction Set

- The availability of different instruction formats in a single computer provided maximum flexibility for the user and minimum waste in memory.
- Variable length instructions were commonly available to utilize the index mode of addressing.
- Instruction types usually offered were: register-to-register, register-to-index, register-to-storage, storage-to-immediate, and storage-to-storage.

CONCLUSION

By comparing the PDP system changes with the general architectual trends, DEC's evolution of architectural structures clearly implemented current technology on a timely basis. Throughout the lives of the PDP-8 and PDP-11 families in particular, DEC addressed limitations and inefficiencies. System performance continually improved, especially processing speeds, operational efficiency, and memory capacities. The architecture was modified to provide these enhancements. DEC has, through the development of the PDP minicomputers, been the single greatest influence on the minicomputer industry. The tremendous success of these low-cost, small-scale systems has, to a large extent, caused the current computer revolution.

BIBLIOGRAPHY

Introduction to Programming, Digital Equipment Corporation, 1970.

PDP-11 Processor Handbook, DEC, 1970.

PDP-11 Processor Handbook, DEC, 1975.

PDP-11 Processor Handbook, DEC, 1980.

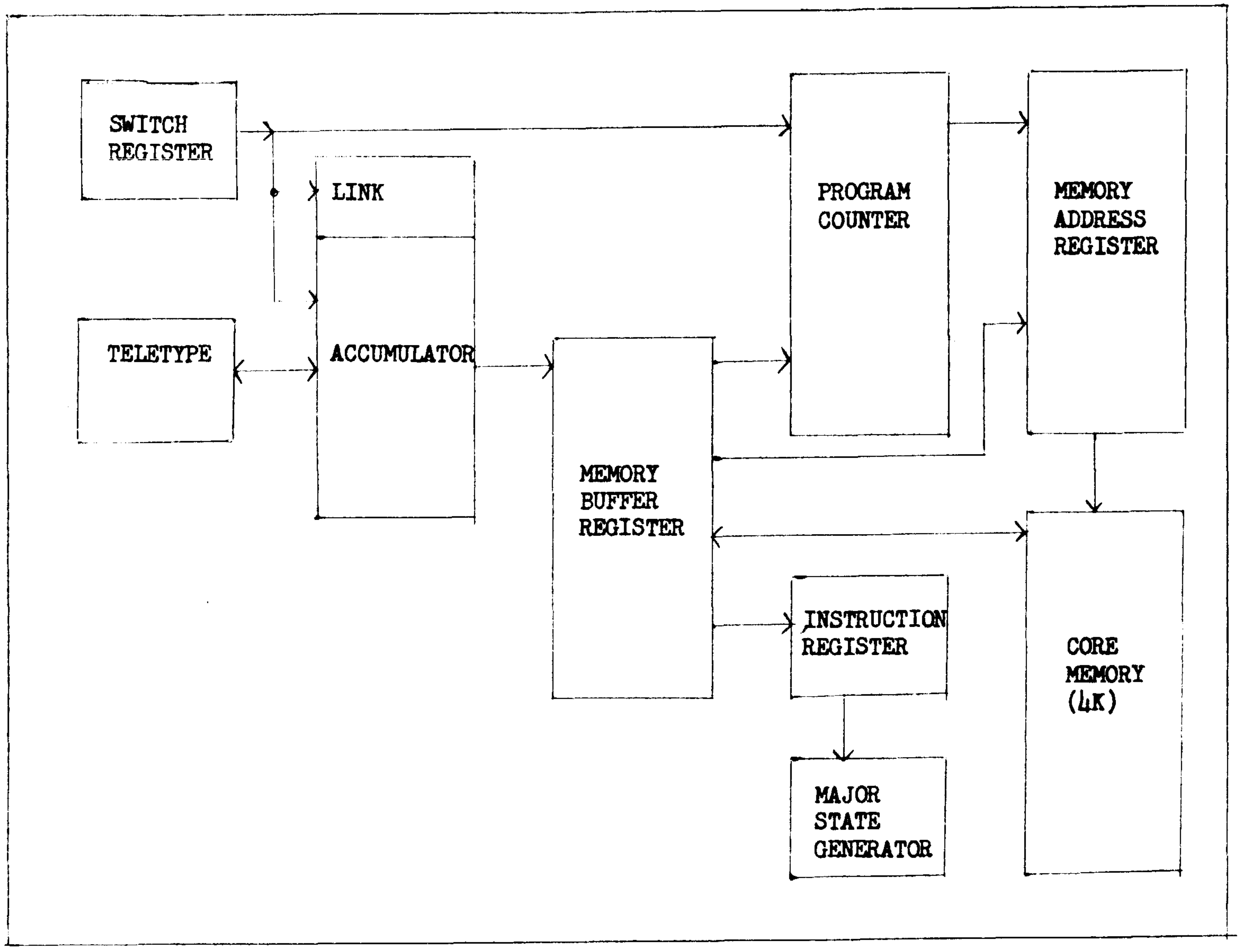
Small Computer Handbook, DEC, 1965.

Small Computer Handbook, DEC, 1968.

Small Computer Handbook, DEC, 1970.

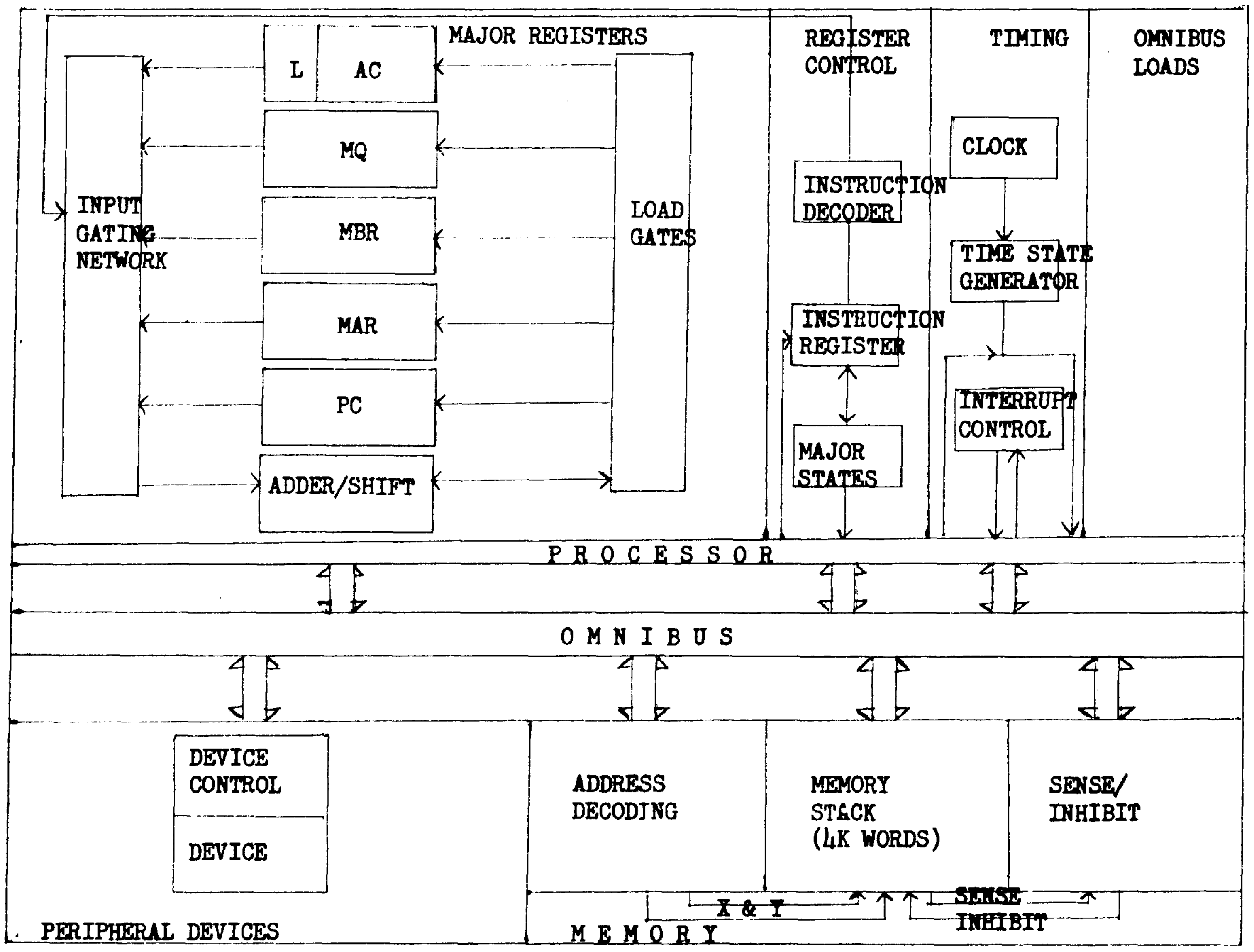
Small Computer Handbook, DEC, 1972.

A P P E N D I X

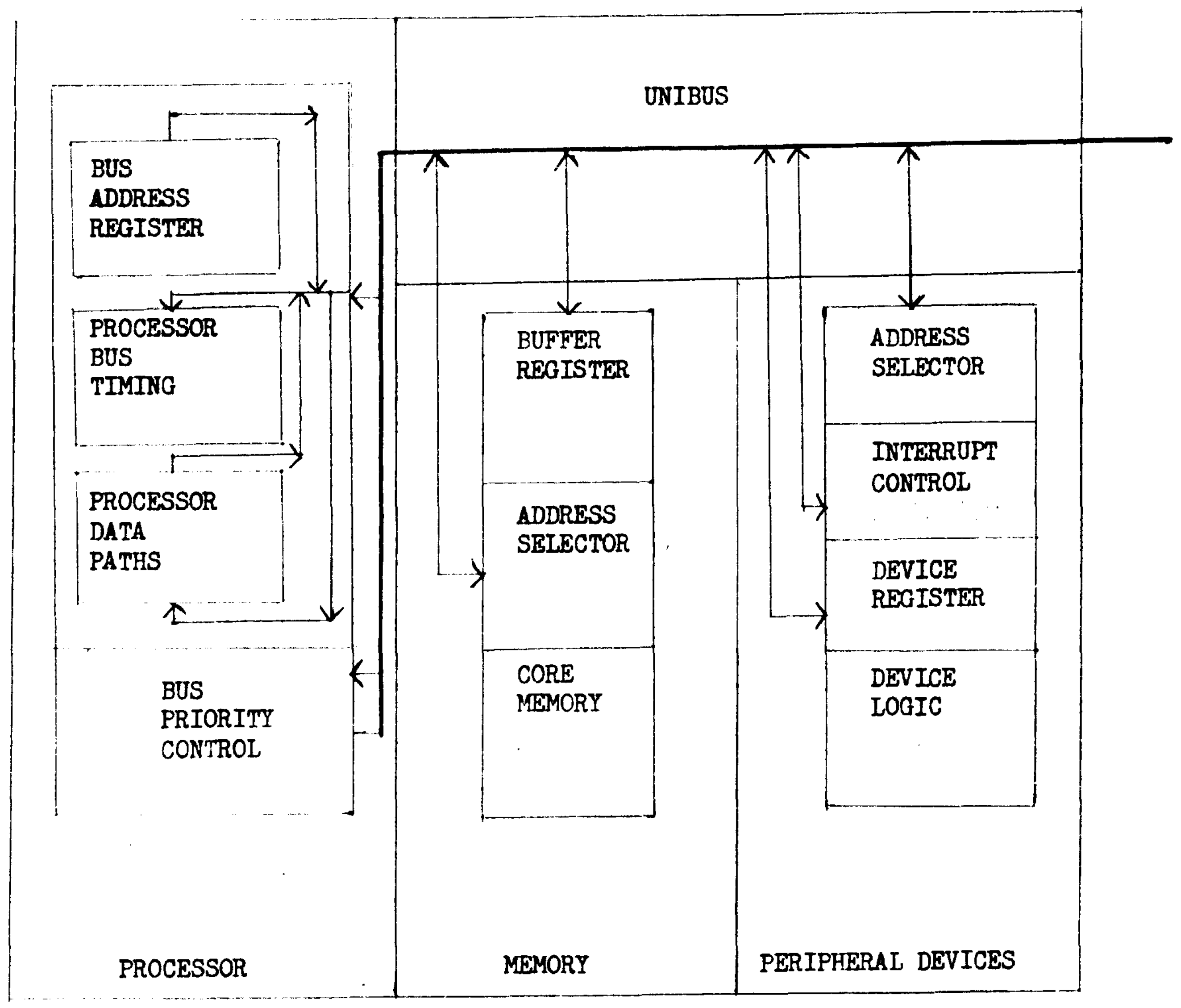


PDP-8 BLOCK DIAGRAM - 1965

APPENDIX 1a



PDP-8/E BLOCK DIAGRAM - 1970



PDP-11 BLOCK DIAGRAM

APPENDIX 2

ARITHMETIC/LOGICAL SPEEDS
(in microseconds)

		ADD	SUBTRACT	MULTIPLY	DIVIDE
PDP-8					
	1965	3.00	6.00	315.00	444.00
	1970	2.60	5.00	256.50	342.40
PDP-11					
	1970	2.30	2.30	NA	NA
	1980	0.30-1.20	0.30-1.20	6.12	7.65

APPENDIX 3

PDP-11 TRAP INFORMATION

Trap Priorities (from highest to lowest)

1. Bus Errors
2. Instruction Traps
3. Trace Trap
4. Stack Overflow Trap
5. Power Failure Trap

Vector Addresses and Trap Errors

000	Reserved
004	CPU Errors
010	Illegal and reserved instructions
014	Breakpoint Trap
020	Input/Output Trap
024	Powerfail
030	Emulator Trap
034	TRAP instruction

APPENDIX 4

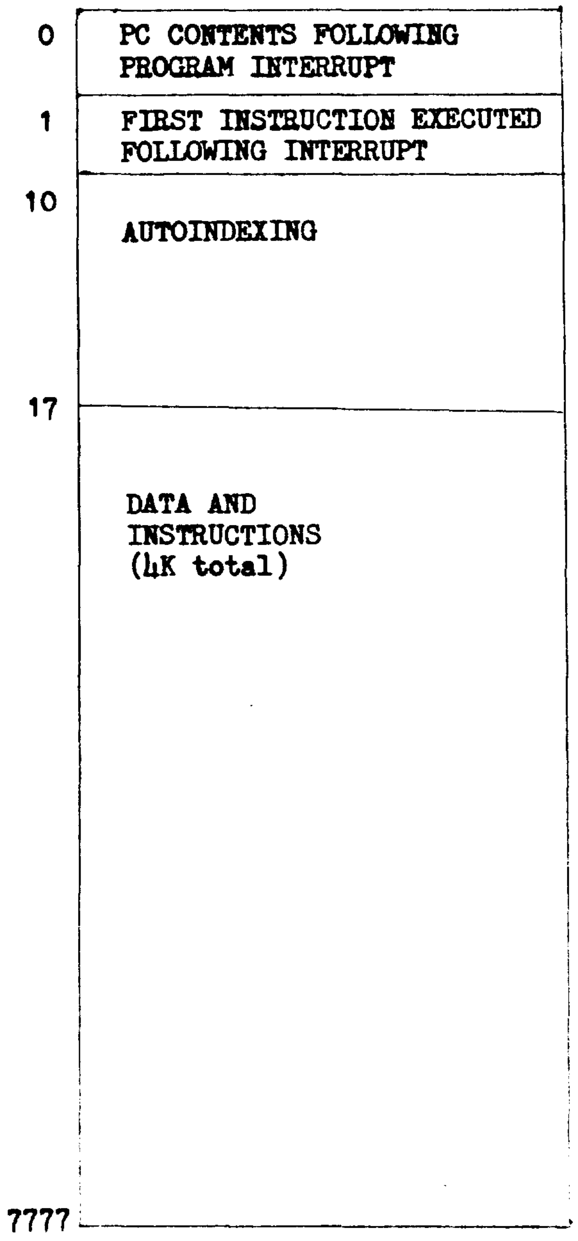
MEMORY CYCLE SPEEDS
(in microseconds)

	Core	MOS	Cache
1965	1.50	-	-
1970	1.20	-	-
1980	0.97	0.51	0.33

*NOTE: Each major state used 1 memory cycle. Instruction Execution Time was the sum of states(cycle time) required.

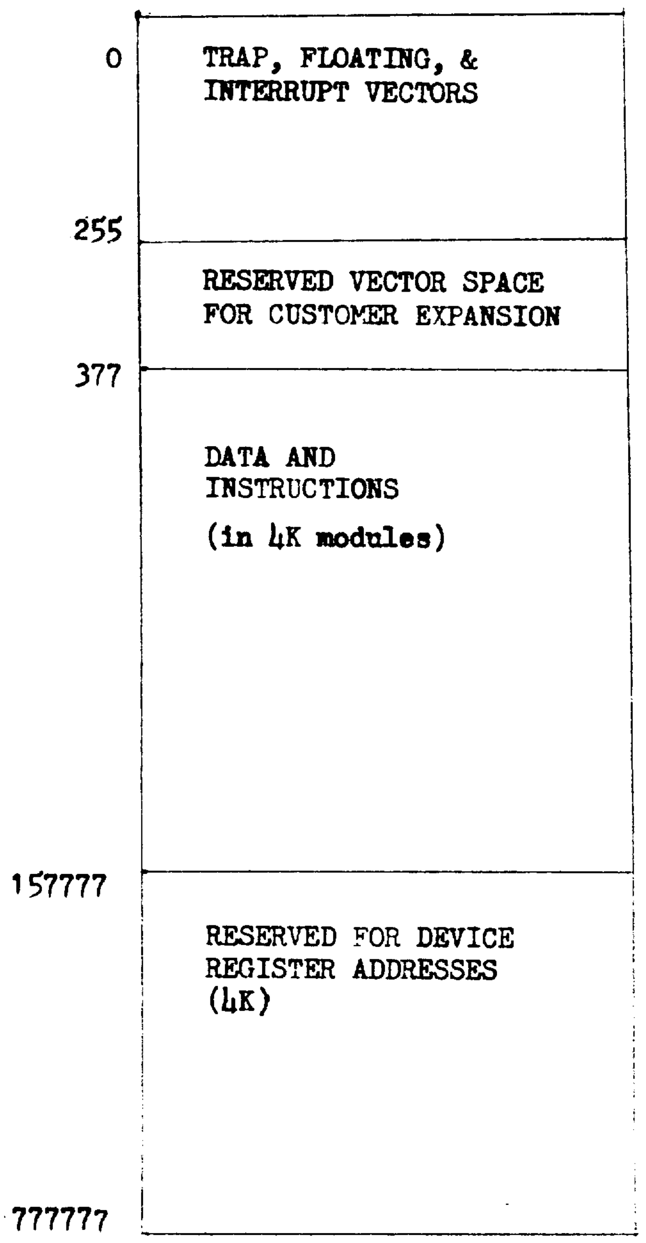
APPENDIX 5a

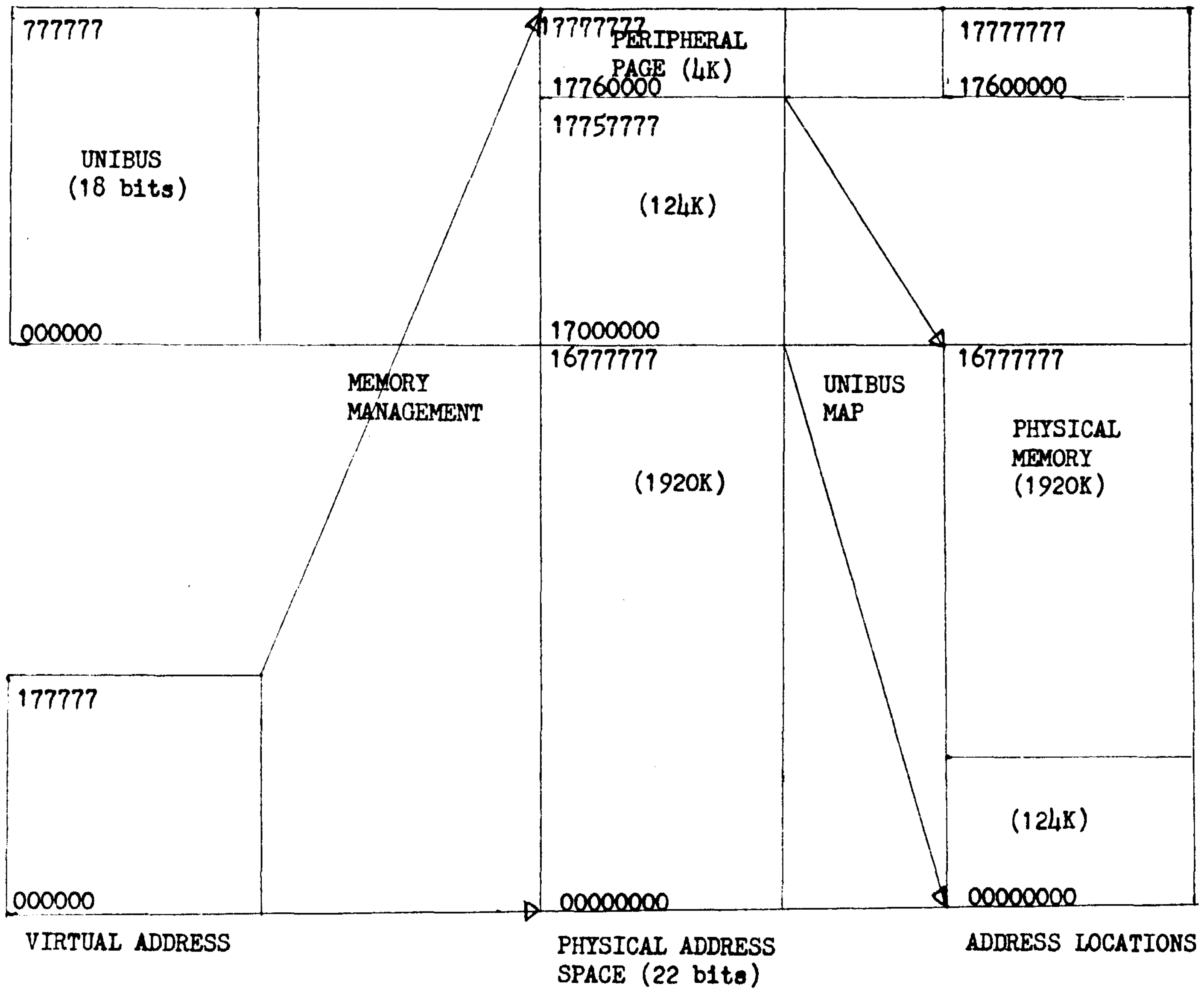
PDP-8 ADDRESS MAP


0
PC CONTENTS FOLLOWING PROGRAM INTERRUPT
1
FIRST INSTRUCTION EXECUTED FOLLOWING INTERRUPT
10
AUTOINDEXING
17
DATA AND INSTRUCTIONS (4K total)
7777

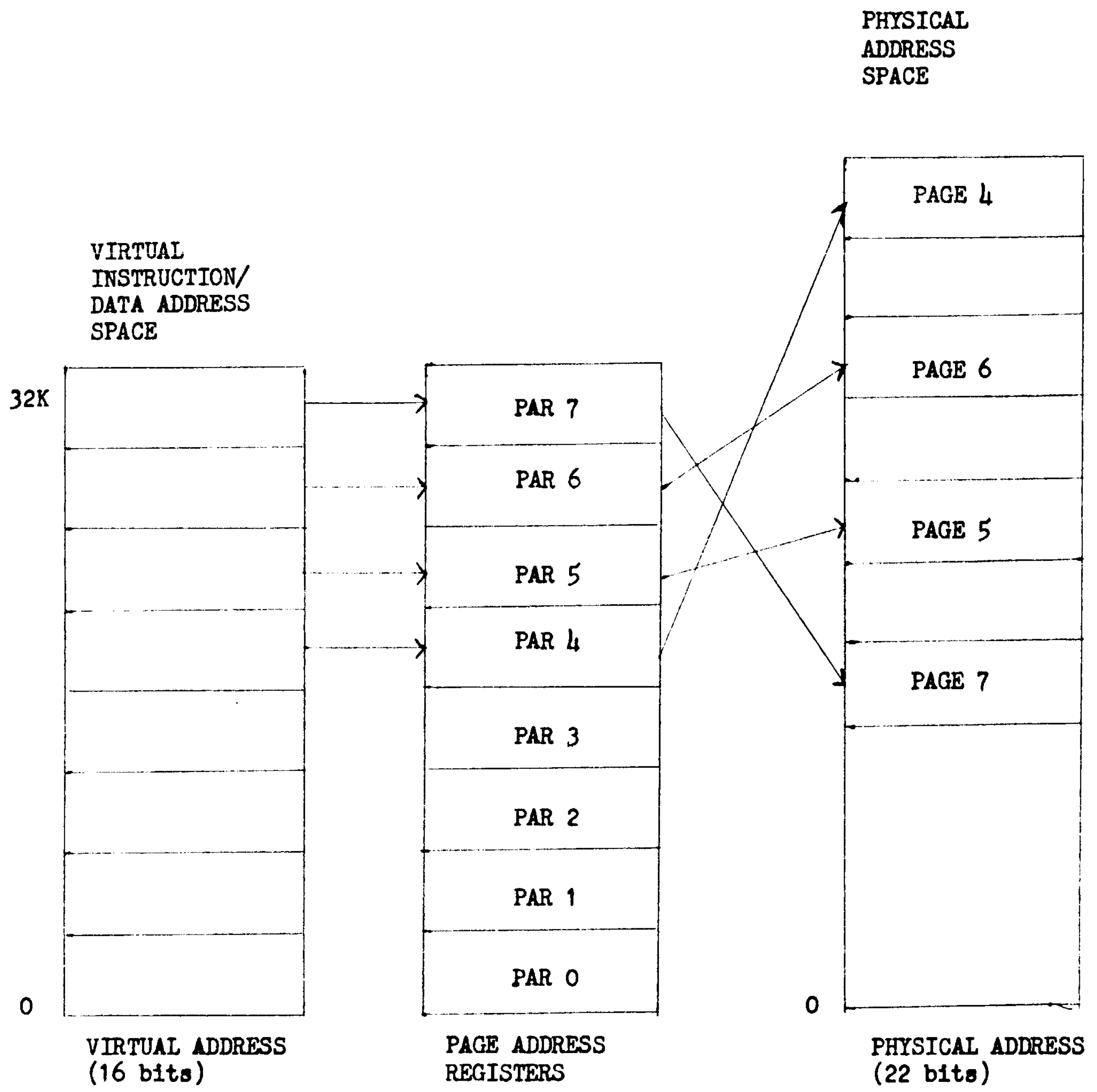
APPENDIX 5b

PDP-11 ADDRESS MAP


0
TRAP, FLOATING, & INTERRUPT VECTORS
255
RESERVED VECTOR SPACE FOR CUSTOMER EXPANSION
377
DATA AND INSTRUCTIONS
(in 4K modules)
157777
RESERVED FOR DEVICE REGISTER ADDRESSES (4K)
777777

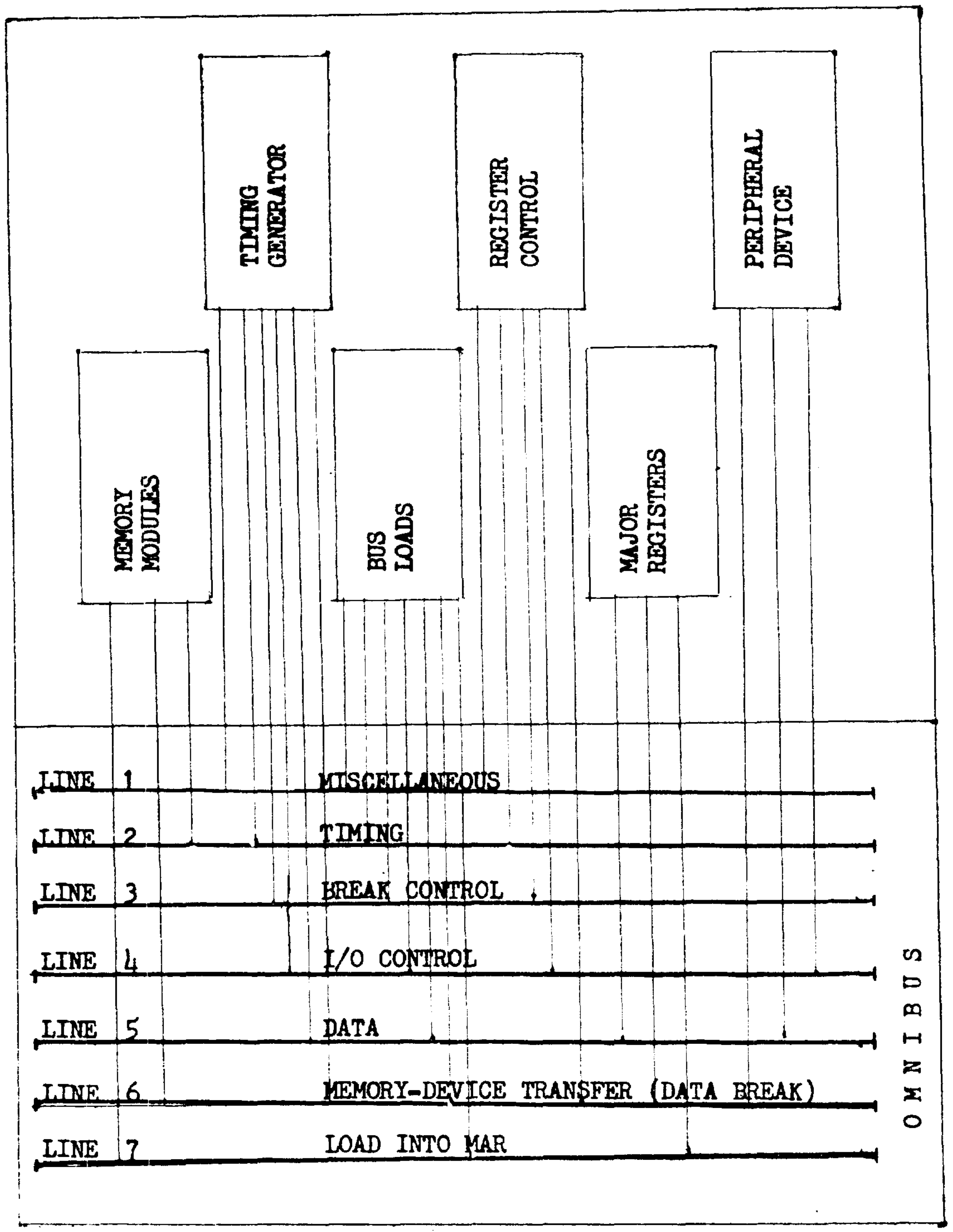


22-BIT MAPPING



VIRTUAL ADDRESS MAPPING ONTO PHYSICAL ADDRESS

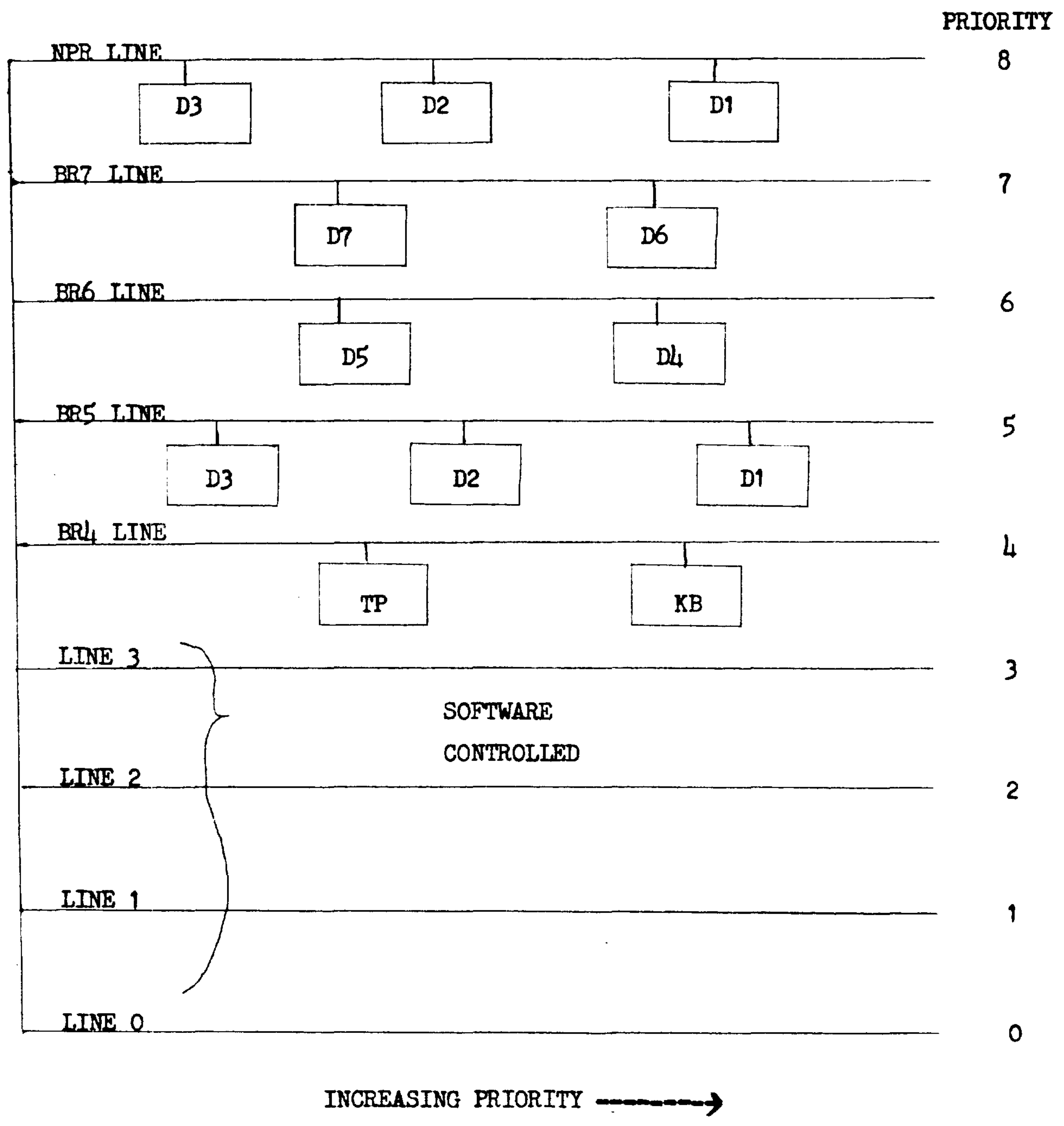
APPENDIX 8a



OMNIBUS (1970)

APPENDIX 8b

UNIBUS



APPENDIX 9a

PDP-8 I/O DEVICE TRANSFER RATES
(in characters per second)

Device	Input Rate	Output Rate
Teletype	10	10
High-Speed Paper Tape Reader	300	--
High-Speed Paper Tape Punch	--	50
Random Access Disk	125,000	125,000
Magnetic Tape	33,000	33,000
Magnetic Drum	30,000	30,000
Line Printer	--	665 - 2,200 (300-1,000 lines per minute)
Card Reader	133	--

APPENDIX 9b

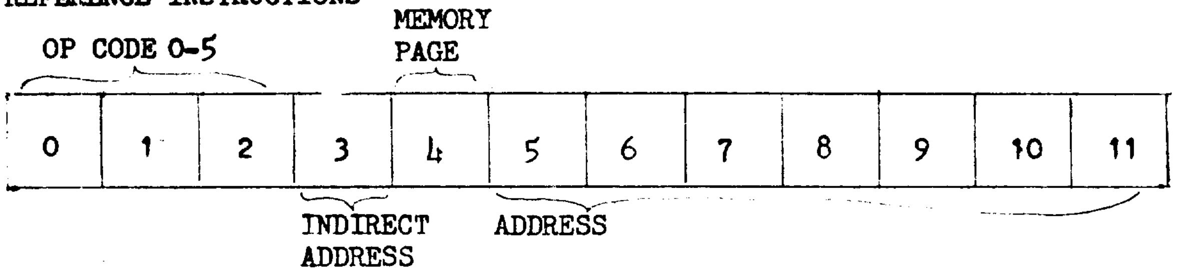
PDP-11 I/O DEVICE TRANSFER RATES
(in characters per second)

Device	Input Rate	Output Rate
Video Display	2 - 5	30 - 1,200
Teleprinter	2 - 5	10 - 30
Disk Pack	300,000 - 2,000,000	300,000 - 2,000,000
Flexible Disk	30,000 - 60,000	30,000 - 60,000
Magnetic Tape	100,000 - 1,250,000	100,000 - 1,250,000
Line Printer	--	665 - 6,650 (300-3,000 lines per minute)
Microfilm	--	22,000 - 66,500 (10,000-30,000 lines per minute)

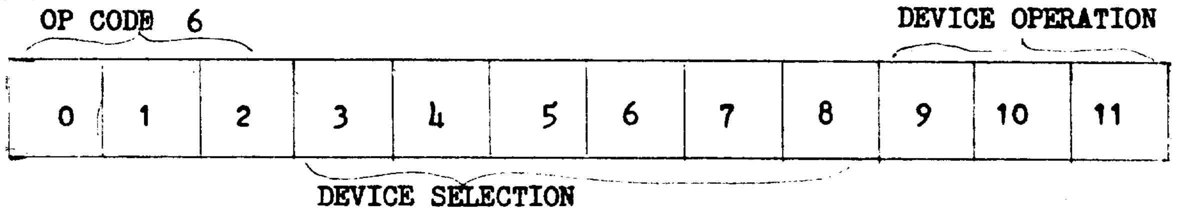
APPENDIX 10a

PDP-8 INSTRUCTION FORMATS

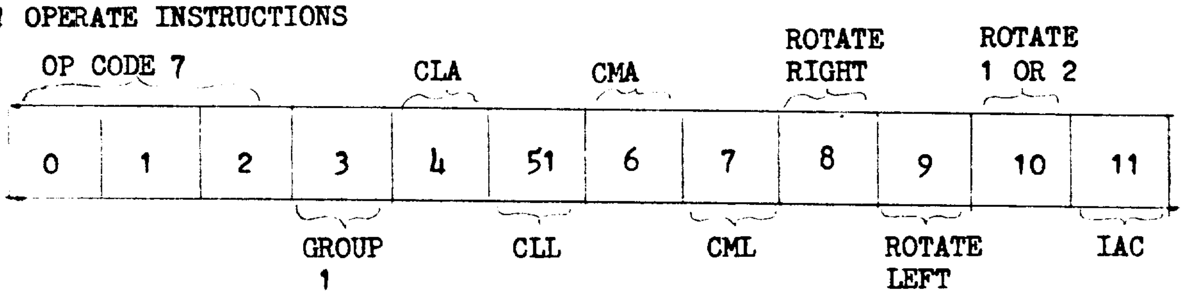
MEMORY REFERENCE INSTRUCTIONS



I/O TRANSFER INSTRUCTIONS



GROUP 1 OPERATE INSTRUCTIONS



<u>KEY:</u>

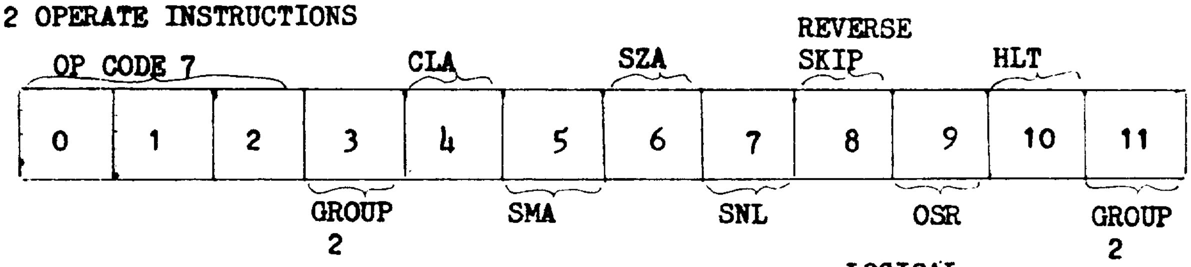
CLA	Clear Accumulator
CLL	Clear Link
CMA	Complement Accumulator
CML	Complement Link
RAR	Rotate Accumulator & Link Right
RAL	Rotate Accumulator & Link Left
RTR	Rotate Accumulator & Link Right Twice
RTL	Rotate Accumulator & Link Left Twice
IAC	Increment Accumulator
NOP	No Operation (bits 3-11 cleared)
STA	Set Accumulator (all bits set)

LOGICAL SEQUENCE:

1 - CLA,CLL
2 - CMA,CML
3 - IAC
4 - RAR,RAL,RTR, RTL

PDP-8 INSTRUCTION FORMATS (cont.)

GROUP 2 OPERATE INSTRUCTIONS



<u>KEY:</u>

CLA -	Clear Accumulator
SMA -	Skip on Minus Accumulator
SZA -	Skip on Zero Accumulator
SNL -	Skip on Non-Zero Link
SKP -	Skip Unconditional (bit 8=1)
SZL -	Skip on Zero Link (bits 7,8=1)
SNA -	Skip on Non-Zero Accumulator (bits 6,8=1)
SPA -	Skip on Positive Accumulator (bits 5,8=1)
OSR -	Or with Switch Register
HLT -	Halt

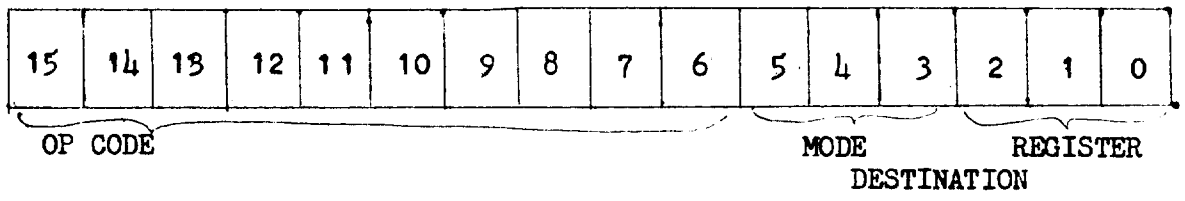
LOGICAL <u>SEQUENCE:</u>

1 - SMA,SZA,SNL
    SPA,SNA,SZL
2 - CLA
3 - OSR
4 - HLT

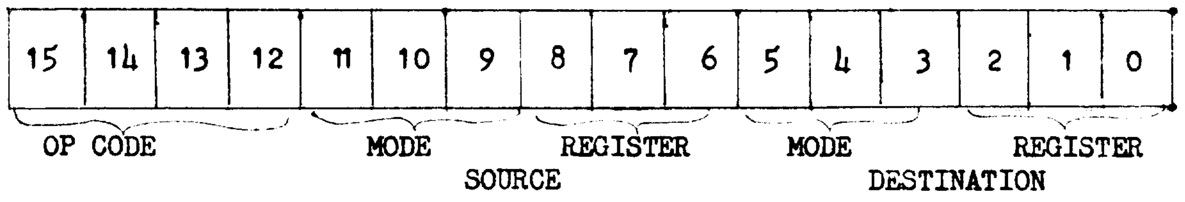
APPENDIX 10b

PDP-11 INSTRUCTION FORMATS

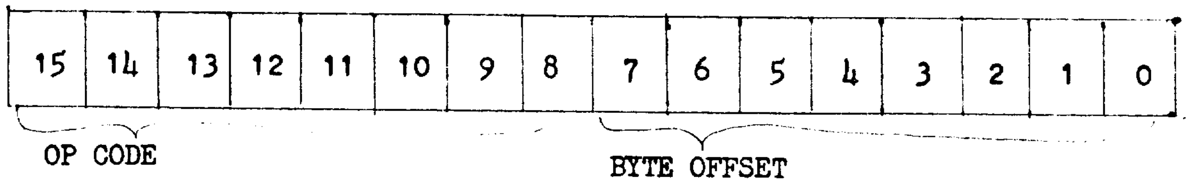
SINGLE OPERAND INSTRUCTIONS



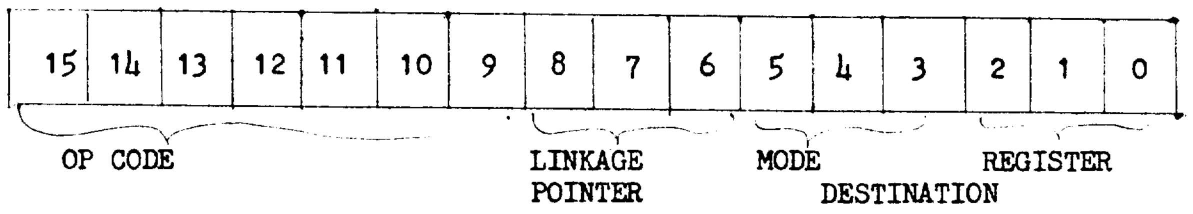
DOUBLE OPERAND INSTRUCTIONS



BRANCH INSTUCTIONS

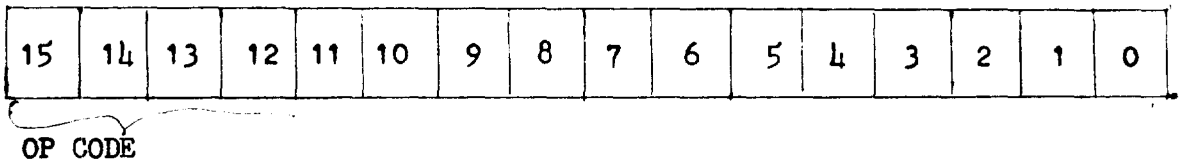


SUBROUTINE INSTRUCTIONS

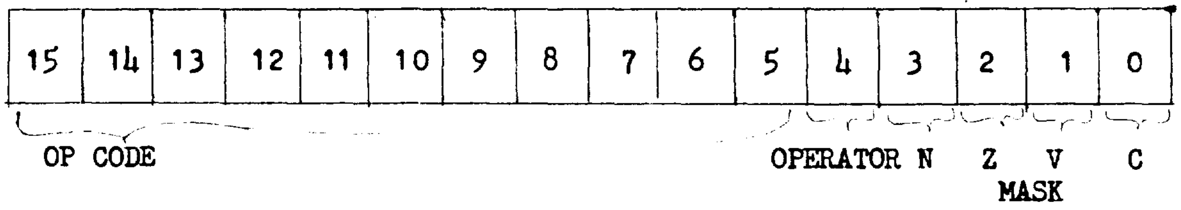


PDP-11 INSTRUCTION FORMATS (cont.)

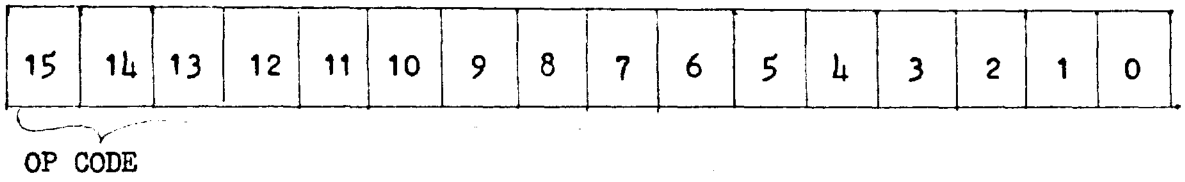
OPERATE INSTUCTIONS



CONDITION CODE INSTRUCTIONS



TRAP INSTRUCTIONS


15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
OP CODE